Hangman Project

Prerequisites:

- variables, conditionals/booleans, loops, arrays

This is a good project to assign to pairs, and to build up in complexity over time. It's also nice because it can involve a lot of interaction and discussion.

Depending on your students and their level, you could combine these parts, and leave more of the implementation figuring out to them.

Part 1: Introducing Hangman

- Start off by playing hangman on the board, asking the students to guess the letters.
- Discuss what the visual components of the game are (the hangman, the letter dashes, the box of used letters).
- Pair off the students and have them create the visual components. It's nice to start off with the visual, to motivate them to keep going and make it functional.

Part 2: Displaying Guessed Letters

- Play the game again with the students.
- Ask the students what variables they might use to represent the word to guess (an array or a string - array preferred, to give them more array practice).
- Ask students how they would check if a letter was inside the word (a for loop iterating over it, with an equality condition).
- Remind students how the keyPressed function works in ProcessingJS, and give them this starter code:

```
var word = ["p", "i", "l", "l", "o", "w"];

keyPressed = function() {
  var letter = key.toString();
  println("You typed " + letter);
};
```

- In their pairs, students now work on displaying the guessed letters above the dashes.

Part 3: Displaying Wrong Letters

- Play the game again with the students.
- Emphasize what the thinking process is for *wrong* letters- "I check every letter in the word, and then finally, when I'm through with the word, I realize that I never found a matching letter, so it must be a wrong guess! Then, I display it in the box."
- Ask them for ideas of how they could keep track of whether they ever found a matching letter (setting a boolean before the loop to false, true if match found, then checking it after the for loop). Note: This may not occur to most students, though it's a common

technique for programmers.
- Ask them for ideas of how they can display each letter spaced out in the box, so they don't all pile on top of each other (a counter of number of wrong letters).
- In pairs, students now work on displaying wrong letters.

Part 4: The Win/Lose States

- Play the game again with the students.
- Emphasize the thinking process for the win state. Every time a student guesses a right letter, say "now I'm going to decide if you've won yet- there are still spaces in the word, so nope!". Every time a student guesses a wrong letter, say "now I'm going to decide if you've lost yet- I still haven't completed the man yet, so nope!"
- Ask students for ideas of how to do the win/lose states in code (tracking variables like numFound/num Wrong and comparing them is a simple way)
- Reply the game using numFound/num Wrong variables, incrementing them each time, and computing your win state aloud that ways.
- In pairs, students now work on implementing win/lose states. Encourage them to have fun with the graphics of them!

Part 5: Edge Cases

- Play the game again with the students.
- Keep track of numFound/num Wrong this time. Plant someone that guesses the same letter twice, or say "what would happen if one of you guessed the same letter?"
- Point out how the numFound/num Wrong logic will now break, and you'll either win or lose earlier than you should.
- Introduce the concept of an "edge case" - a situation that doesn't happen all the time, but when it does, it breaks the code horribly. Talk about famous edge cases, like the Y2K bug.
- Ask students for ideas of how they could keep track of guessed letters (e.g. a letters Guessed array, checking indexOf and returning, pushing the new letter into it)
- In pairs, students work on implementing the guessed letters check.

Part 6: Artificial Intelligence

- Play the game again with the students, but divide them into 2 teams, with gallows for each.
- Suggest "what if we wanted a 2-player game, but we didn't have a second player? We'd have to implement a computer player."
- Ask them what games they've played with computer players.
- Introduce the concept of "AI" - artificial intelligence, used by computer players to approximate what humans would do in that situation.
- Ask them if they've ever heard of "AI" - like in movies (there are lots!)
- Ask them for ideas of how to implement a hangman AI. There are lots of cool ways they could come up with. To name a few:
    - Guessing the most common letters first.

        - When they suggest this, ask how they might come up with a ranked common letters list (like by going through a dictionary and tracking frequency)
    - Guessing based on common letter combinations.
    - Guessing vowels first.
    - Comparing to words in a dictionary, and guessing based on most probable word.
- Talk about combining strategies, using different ones as the game progresses.
- Ask how they would implement an easy vs. hard hangman AI (“what’s the dumbest AI we could make?”)
- Encourage them to read “Final Jeopardy” and “The Most Human Human” if they’re interested in learning more about AI.
- No programming required for this part, unless your students are savvy enough to implement an AI (like making an array of sorted letters and guessing from there).
- Remember to have fun!